Addressing Modularity for Heterogeneous Multi-model Systems using Model Federation

Fahad Rafique Golra, Antoine Beugnard, Fabien Dagnat, Sylvain Guerin, Christophe Guychard

To cite this version:
Addressing Modularity for Heterogeneous Multi-model Systems using Model Federation

Fahad R. Golra  Antoine Beugnard  Fabien Dagnat  Sylvain Guerin  Christophe Guychard
IRISA / Télécom Bretagne, Institut Mines-Telecom  Openflexo
Brest, France  Brest, France
fahad.golra, antoine.beugnard, fabien.dagnat, sylvain.guerin, christophe.guychard @openflexo.org

1. Introduction

Models form the core of software development processes in the context of Model Driven Engineering (MDE). This increasingly important role of modeling has led to a plethora of domain specific modeling languages (DSLs/DSMLs) (Mernik et al. 2005). Each DSL captures the specific knowledge involving a certain domain to provide adapted features. This specialization contributes to improved efficiency and enhanced quality of systems. The design of complex software-intensive systems might require several DSLs to handle various concerns. Heterogeneity of models in the modeling space leads to multiple issues for synchronizing models, maintaining traceability and managing global consistency of the system. Complex software intensive systems usually need to integrate the viewpoints of various stakeholders and system experts (Tyree and Akerman 2005). These viewpoints of different stakeholders might be related to multiple models, each taking care of a specific concern. Adding those cross concerns to the equation makes modeling of these systems even more difficult. One way to handle this complexity is to improve the modularity of the modeling space that already seems modular because of multiple heterogeneous models.

Considering this situation, current approaches are typically oriented towards translation/model transformations to bring all DSLs under one operating paradigm (Eclipse 2016, Jezéquel et al. 2011), composition of (meta)models (France et al. 2007, Eker et al. 2003, Hardebolle and Boulanger 2008, Kong and Alexander 2003, Berg and Möller-Pedersen 2015) and unification through intermediate languages (Vernadat 2002). All these endeavors aim at providing a single language, (meta)model, modeling paradigm to cover all heterogeneous concerns that one can come across in the development of a complex software system. We argue that these DSLs should be kept in their own paradigms and modeling element redundancies should not be created by transforming them to a single paradigm through integration, composition or unification. A better approach would be to create bridges between these distinct paradigms and maintain a dynamic link so that these different models can be synchronized for global consistency, traced from one another and used for the development of cross-concern viewpoints/models.

It might seem that our view of modularity is the antithesis of separation of concerns, which advocates loose coupling between the modules. In reality, taking care of the cross-cutting concerns in multiple models of a modeling space ensures a better modularization, much like aspect oriented modeling (Masuhara and Kiczales 2003). These models can have considerable overlapping features that need to be synchronized for the global consistency. Multi-Dimensional Separation of concerns (MDSOC) defines modularity in multi-dimensional spaces of units, where dimensions represent different modularizations (Tarr et al. 1999). We consider two dimensions of modularity for our proposal. The structural modularity places all features, primarily serving for a common concern, in a single model (a module). The conceptual modularity looks for the projections of secondary concerns of these features in conceptual space and creates modules out of the overlapping or tightly coupled features from different models. A multi-model system is structurally well modularized because each model serves a specific concern. However, the overlapping features and tight semantic coupling between them in the context of heterogeneous models, call...
for conceptual modularity as well. For the system to work as a whole, well-defined interfaces should guarantee the contracts between these modules. By modularizing complex software-intensive systems in both these dimensions, we expect strong impacts on the reduction of complexity for modeling heterogeneous systems. We believe that our approach will favor reuse, synchronization, flexibility, maintainability and conceptual alignment of existing/legacy (meta)models in the modeling space.

The rest of this paper is organized as follows. First, we present our approach of model federation in Section 2. Then, in Section 3 we explain the proposed approach in the context of modularity for the design of multi-models systems that face the problem of heterogeneity. Then, Section 4 discusses the state of the art in multi-paradigm modeling in the context of modularity. Finally, we conclude this paper in Section 5.

2. Model Federation

Software Industry needs methods for seamless integration of specialized formalisms and tools that they use for the development of complex software systems. These specialized formalisms, tools and DSLs used in the development process add to the heterogeneity of modeling space. These models need to be synchronized for the global consistency of the system. Synchronization of different models in the modeling space is handled through the co-evolution techniques (Hebig et al. 2013). However when these models belong to different paradigms, their co-evolution becomes difficult. Before going into the details of how we propose to deal with these issues, let’s take a look at the concepts of model federation.

Model federation provides a mechanism for the integration of heterogeneous models. This integration of models may serve to develop new cross-concern viewpoints/models from several existing models or to synchronize the models that are used in the modeling space for designing a system. We propose to keep heterogeneous models in their own paradigms. Hence, we divide the modeling space in two distinct spaces i.e. the technological space and the conceptual space, as shown in the Figure 1. Technological spaces are defined environments that allow heterogeneous models to use their own paradigm, formalism and tooling within their private modeling space. For the design of a complex system, a conceptual space may be surrounded by many technological spaces. Each technological space contains the models that follow a specific modeling paradigm to serve specific concerns. A conceptual space is where a set of models can be federated to develop a new cross-concern viewpoint/model, called a virtual model. Virtual models are made up of features, which are reused from the models of technological space. New features can also be developed for the virtual model, which do not belong to any of the existing technological spaces. Each of these features, called a flexo concept, serves as a building block for a virtual model. A virtual model is responsible for managing the life cycle of the flexo concepts that it contains. A virtual model can be serialized back into a new or existing technological space for the development of a complex system.

The virtual models in the conceptual space follow the formalisms defined by our methodology. We have developed a generic modeling language to define virtual models. Technological models follow their own paradigm or formalisms depending on their specific technological space. A virtual model can not access the features of a technological model, unless it knows what paradigm or formalism it is following. A connection is needed between these spaces for making the features of a technological model available in the conceptual space. A connection between a specific technological space and the conceptual space is realized using technological connectors. These connectors allow access for reading, writing and eventually synchronizing the information between virtual models and the technological models.

Once a technological connector is available for a specific technological space, all the models in that space and their respective elements become accessible from within the conceptual space. A virtual model is composed of the concepts precisely developed for it and the ones reused from other technological models. The concepts specifically developed for the virtual model are the local flexo concepts. The concepts from a technological model can be reused in two different ways. First, the designer can create a dynamic link between a local flexo concept and the technological model concept that it wants to use, as shown in virtual model A of Figure 1. Once created, this connection is maintained to synchronize any further changes. Second, the designer can also choose to create a local ‘proxy’ of that concept in the virtual model by translating it into a remote flexo concept. A remote flexo concept, as depicted in virtual model B of Figure 1, is a proxy of the technological model concept and thus introduces some redundancy. We suggest and prefer to use the first method of maintaining a dynamic link. Even in the second method, a link is maintained between the local proxy and the remote feature to ensure their synchronization.

The connection between the conceptual and technological spaces is bi-directional. A virtual model in the conceptual space can be updated when a corresponding technological model is evolved and conversely, a technological model can also be updated, once the corresponding virtual model is modified. The mechanism for updating the information in both directions is kept flexible. The designer can choose to get notified when a corresponding model is evolved or to get automatically synchronized, depending on the semantics defined in the technological connector.

A model slot is the modeling element that realizes a connection between a virtual model and a technological model. As already explained, technology connectors connect the conceptual space with a specific technological space. These connectors allow the creation of a dynamic link (using model slots) between a specific virtual model and a specific technological model of the connected technological space. If a virtual model is considered as a component, model slot serves as its interface. The model slot exposes a view on the structural and behavioral contents of the technological model to the virtual model. Once a virtual model is connected, it can read/modify the attributes using roles and execute the actions using edition-Action. Roles and editionActions are to a flexo concept, what attributes and methods are to a class, except that roles and editionAc-

---

**Figure 1. Modeling space in Openflexo**

Legend:
- Model
- Technological connector
- Flexo concept (local)
- Model slot
- Flexo concept (remote)


Figure 2. Mapping between concepts

3. Modeling Space Modularity

A modeling space with heterogeneous models seems to be modular, because each model affects a distinct concern for the system under development. Our methodology places all those heterogeneous models in multiple technological spaces. Each technological space adheres to a specific paradigm, so a technological space may end up containing multiple models. A separate conceptual space allows the development of new models. Figure 3 illustrates four different perspectives from a scenario that we are going to use as a running example to explain the key notions of our methodology. In Figure 3a, a technological model for missile system and another for a radar system are placed in their respective technological spaces TS-M and TS-R. Placement of both these models in separate technological spaces illustrates that they use different paradigms. A virtual model for the development of a ship is under-development in the conceptual space, CS. To this point our modeling space is modular, all three models are serving their specific concerns and it does not really matter if they belong to different paradigms. We argue that this modularity of the modeling space is structural i.e. all related modeling elements are placed in a single model to take care of a particular concern. This specific concern may be quite different from other concerns, and thus a specialized modeling language, a DSL, might be required. So let's consider that the models of radar system and missile system are developed in two distinct DSLs. The user wants to develop a model for ship, which shall use the modeling language described by our language. Once developed, it is possible to serialize the ship model to any other technological space.

Development of some specialized views of the system or new models might need to access multiple modeling elements from different existing models belonging to distinct technological spaces. These new views/models are created as virtual models in the conceptual space and the technological models are considered as their resources. As resources, they allow the virtual models to extract information from them. Connecting information from varying resources allows the generation of new meaningful concepts. Modeling the concepts, whose building blocks are coming from multiple resources, needs to "cross-cut" the existing heterogeneous models. This seems to create additional couplings between the modular architecture of modeling space with models serving as distinct modules. In reality, these couplings between the modules already exist at a conceptual level, they are just reified in our approach. These existing semantic links are shown in Figure 3b. Using multi-dimensional separation of concerns, we propose to use structural and conceptual dimensions for modularity. Figure 3c depicts a structural modularization of the modeling space. Structural modularization means that each modeling element is placed in a single structure, a model in our case, to achieve its primary concern. The other concerns of a modeling element can be projected to the conceptual dimension, which are used to identify meaningful semantic links between the modeling elements of different models. These semantic links might, for example, represent that two modeling elements from models following different paradigms are conceptually identical. In our scenario, the modeling element representing weight in TS-M and TS-R is identical, as shown in Figure 3b. Using such conceptual links between different modeling elements, our methodology can ensure a conceptual alignment of their respective models. Some semantic links between modeling elements of different paradigms might not make sense in their respective technological spaces. However combining those elements in a common conceptual space might yield new concepts. For example, the modeling element representing a missile in TS-M when linked to the modeling element representing radar in TS-R can help develop a model for a battleship that uses both missiles and radars. Figure 3d depicts this case, where the conceptual space is in need of these two concepts and a third concept of weight which is identical between the two technological models.

We like the definition of modularity as, "a continuum describing the degree to which a system’s components can be separated and recombined" (Schilling 2000). It has two main considerations: 1) the partition of a system into smaller modules which serve specific concerns and, 2) to establish a interaction contract between them, so that they can serve as one whole. Figure 3b shows our scenario where our virtual model can be developed by reusing the concepts missile, radar and weight in TS-M and TS-R. This figure shows the conceptual projection of TS-M and TS-R on the CS. The features of the virtual model, depicted in grey, are the conceptual projections of the model elements that are still placed in their respective technological spaces. This conceptual projection means that the features of radar and missile models continue to serve their primary concern in their respective models. However their projection on the virtual model allows to use the secondary concerns of those features. It seems as if instead of modularizing the system, the proposed methodology is actually combining it. The actual architecture of the system after conceptual modularity is better represented by Figure 3a. Here, the original modules are kept intact and the development of the virtual model in the conceptual space is carried out by linking it to the cross concerns from other models.
The proposed methodology provides mechanisms for establishing a dynamic link between the virtual model and the technological models. These mechanisms are offered by the use of technological connectors and model slots, as explained in the previous section. The overlapping nature of a cross-cutting concern forms an inherent coupling between models, and model federation offers a mean to conceptualize it in a single module, the virtual model.

Figure 3 describes the informal process to use model federation to create mappings between heterogeneous models. The development of a virtual model starts by developing local flexo concepts according to the specifications. This produces a virtual model with all local flexo concepts and needs the cross cutting concerns from other models for its completion, as shown in Figure 4a. Then the concepts needed to complete the virtual model are identified in the available technological spaces, as shown in Figure 4b. Finally, mapping are created between the flexo concepts and the concepts of technological models, based on either dynamic links (Figure 4c-1) or developing remote flexo concepts as proxies for the technological model elements(Figure 4c-2).

Software requirements evolve over time, resulting in model evolution for design and subsequent phases. It becomes hard to maintain multiple heterogeneous models in the modeling space when they are evolving over time. Evolution of a source model results in the evolution of all the target models. When traditional model driven approaches like transformations or model compositions are used, these target models need to be “regenerated”. A typical drawback of model transformation and model composition techniques is that they do not allow independent modifications to individual parts of source/target models. The modeling elements of the output models in these techniques do not maintain any dynamic link with the corresponding elements of input models. Model federation does not adhere to the concepts of source and target models. It rather advocates the synchronization of multiple federated models. In case of evolution, the complete model does not need to be regenerated, only the concerned elements are updated. If a model in a specific technological space evolves over time, changes are synchronized to the virtual model and vice versa. The user remains in control whether to opt for automatic synchronization or to demand an alert-based configuration to remain in control. Openflexo platform takes care of maintaining the dynamic link between the heterogeneous models and their synchronization. Simultaneous changes to common parts of the models at both ends can cause conflicts. For the moment, it is the designers responsibility to resolve such conflicts. However, we are looking forward to develop some mechanism, inspired from version control and content management systems, that can assist the designer for resolving such conflicts.

4. Related Approaches

There are several approaches that address the integration challenges for domain viewpoints while trying to keep the modeling space as modular as possible. With new DSLs coming to prominence, the problem of model heterogeneity is being highlighted more than ever before. The most commonly used approach is the UML profiling mechanism that allows the customization of UML for any specific domain (Schic 2007). Even though this approach is rather easy to implement with case tool customizations using UML profiles, it has its drawbacks. This technique forces us to the world of UML and MOF conformity, which might not be an efficient solution for many heterogeneous models. For example, we have come across models for process engineering that do not adhere to MOF paradigm, and so its impossible to extend them using UML profiles. Furthermore the designer will face the usual problems linked to UML profile mechanisms (France et al. 2006). There are other approaches that deal with model heterogeneity by extending existing metamodels, like EMF facets (Eclipse 2016) and KerMeta (Jezquel et al. 2011), but they pose the same problems of MOF conformity. EMF facets provide a non-intrusive extension mechanism to add additional attributes, references or operations on a model without modifying it or its metamodel i.e. these extensions can be loaded or unloaded on the fly (Eclipse 2016). On the other hand, KerMeta provides the functionality for weaving concerns to existing metamodeling
Figure 4. Model federation process for modularization

- **Legend:**
  - Orange: DSL Missile system
  - Green: Virtual Model
  - Grey: Proxy elements
  - Blue: DSL Radar system
  - Black: Required concepts

(a) Incomplete virtual model  
(b) Identified required concepts

Ts-M  
Ts-R

Legend:

- DSL Missile system
- Virtual Model
- Proxy elements
- DSL Radar system
- Required concepts

A) Incomplete virtual model
b) Identified required concepts
c1) Dynamic links
c2) Remote flexo concepts

5. Conclusions

This paper addresses the issues faced by software designers when heterogeneous models are used during the design phase of complex software-intensive systems. We proposed a modular approach
to handle such issues using model federation. The underlying concepts of model federation were explained, with a particular emphasis on modularity. Modularity of multi-concern systems is presented in two-dimensional perspective of structural modularity and conceptual modularity. Modeling elements are usually combined together to form a complete model that caters for a specific concern, which we call structural modularity. However, the same modeling elements might serve other secondary concerns, in respect to the model that they are placed in. Our approach proposes a mechanism to group these secondary concerns together to form a distinct model serving a particular concern, keeping modularity in mind. Model federation is particularly addressing modularity for heterogeneous modeling spaces, where multiple models come from different technological spaces. A limitation of our approach is the availability of different technological connectors. We have already developed the technological connectors for various paradigms (EMF, OWL, XML/XSD, MS Office document formats, etc.). Associated tooling is implemented as an open source project, Openflexo. This enables the users to develop new technological connectors for their specific needs. We are looking forward to develop a high level language that can replace Java code to type the interfaces of the modules for defining the contracts.

Acknowledgments

This research work is a result of a collaboration between PASS team and Openflexo under FORMOSE project. We are thankful to Agence Nationale de la Recherche, ANR for funding this project.

References


